Weekly Report #6

Group Nr. 21: "Face Reconstruction"

February 15, 2022

1 Weekly Progress

Dataset. While checking the data recorded by the Kinect v2 depth camera last week, we found that its depth images were low resolution. In order to check whether the facial landmarks are well positioned in the right place, we brought data recorded by another depth camera, the Realsense L515, into the pipeline the, which has a higher resolution depth frame of 1280×720. As shown in Figure 1, this is the mesh reconstructed from the Realsense data, and we can see the location of the facial landmarks more clearly.

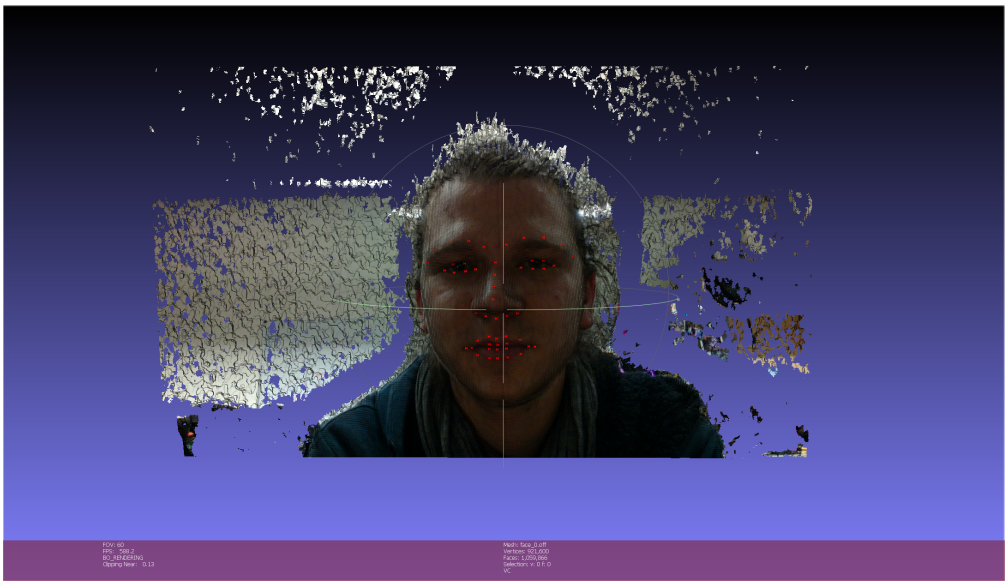

Figure 1: The reconstructed scene mesh from Realsense data

Reading Basel Face Model. For the sake of reading the data of Basel Face Model (2019), since the data is stored in H5 file format, we included the HDF5 library to read the H5 file Besides, we realized that the model has three main parameters, which are **shape**, **color**, and **expression**, where shape and color have 199 principal components, and expression one with 100 of that.

Therefore, we consider these three parameters as `Eigen::VectorXd` vectors of the corresponding dimensions, and our primary task will be to introduce Optimizer to optimize these three vectors so that their shapes are closest to the faces observed in our data sequence.

Moreover, we also write a export function to get the obj mesh of the parametric face model by given three parameters. Figure 7 depicts the mean face exported by this function.

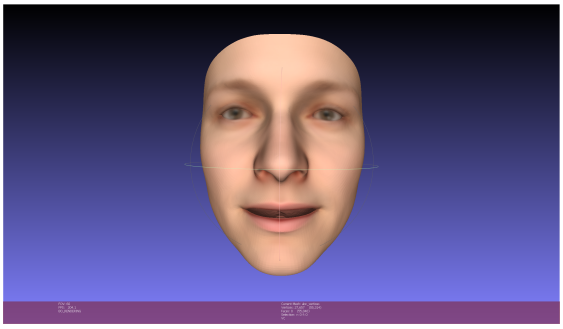

Figure 2: Meanface mesh exported from BFM

Procrustes Alignment. We applied Procrustes analysis to get the initial pose (scale, rotation, translation) of the face. As this is not the final optimisation step, we align with the facial landmarks. However, during our investigation, we found that the landmarks of the face outline are not reliable when they are backprojected to the 3D space. Especially for the landmarks around the chin. They are often backprojected onto the neck, causing the face model to face down after the initial alignment as shown in Fig.4. Therefore, we decided to ignore the landmarks of the face outline (landmark 1 - 17 given by dlib in Fig.3). Fig.5 shows the alignment without landmarks of face outline.

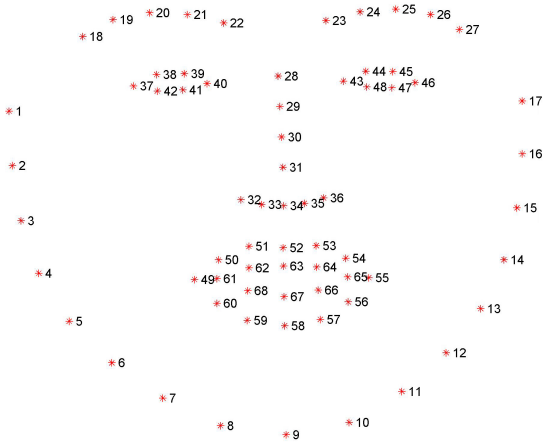

Figure 3: Facial landmarks given by dlib.

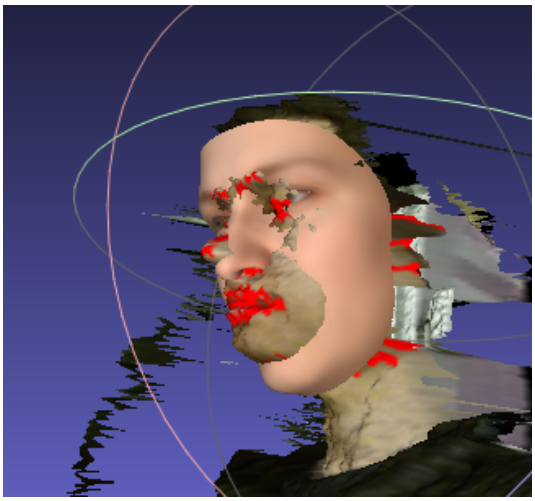

Figure 4: Procrustes Alignment with all 68 landmarks.

Figure 5: Procrustes Alignment with 51 landmarks only (landmarks of face outline removed).

Optimization - Sparse Term. After obtaining a coarser estimate of the pose by the Procrustes algorithm, we pass this pose and the randomly generated **BFMParameters** (aka. the three `VectorXd` vectors representing shape, color and expression mentioned above) into the BFMOptimizer. Firstly, we construct the Ceres cost function of SparseTerm, which we call **LandmarkMatchCost**. For each landmark, we add a ResidualBlock, in which we minimize the difference between the pose transformed position of the landmark of the BFM model and the landmarks of observed position (reconstructed mesh from frames). Using this, we optimize the face shape and expression weights of the Basel Face Model to adapt it to the depth image. We can also optimize the pose in the same step, but we are currently using a transformation matrix and are still thinking about whether we want to constrain the transformation to only Similar Transformations (loc, rot, uniform scale) as opposed to allowing a full optimization of the matrix.

- Tong Yan Chan (03722291): Procrustes alignment, BFM Model Optimizer, Bugs fixing
- Dushyant Anirudhdhabhai Dave (03728740): Left the group
- Daniel Schubert (03666228): Procrustes alignment (scaling), BFM manual landmark placement, BFM Model Optimizer
- Chang Luo (03759570): Make BFMContainer, Read BFM H5 File, Extract frames from Realsense camera by python scripts, BFM Model Optimizer

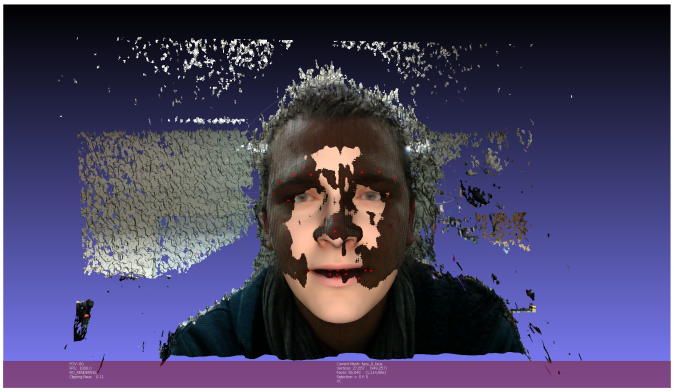

Figure 6: The BFM model and scene mesh after Procrustes Alignment and optimisation of shape and expression

2 Problems

Dataset. At the beginning, we found that the blue channel in the color frames exported by Realsense was always 255, so the whole picture was very blue. We later found out that this was because OpenCV reads the color images using the BGR color mode by default, so we needed to do an extra conversion of the color space, and the problem was quickly solved.

Dependency. When we tried to add the HDF5 library, we ran into many problems, specifically, we had installed HDF5 via vcpkg, but CMake still couldn't recognize its static link library location properly, we wasted a lot of time on this, and finally found that we should add an extra *[cpp]* installation option to make sure it was installed in c++, so that the HDF5 library can be recognized and called correctly.

Landmarks. We had a variety of issues with our data: The 68-landmark-indices did not match for the latest Basel Face Model (2019), which provides 40 landmarks are not in the scope of ibug 300W annotation[1], which is given by landmark detector, so we had to create our own markers. Some of the landmarks created a lot of confusion for the algorithm: The landmarks near the chin are often projected onto the neck during the 3D reconstruction, so the depth position of these landmarks is oftentimes unusable. In the end, we decided to drop these landmarks from our analysis and focus on the other (index start from 18 according to ibug 300 annotation [1]).

Cost Scale Regularization. One problem that occured was that we got a lot of overfitting of the face model which lead to "interesting" results. To solve this, we added a regularization term to the optimization problem. This brought up another issue where we noticed that the scale of our cost function is all over the place: While the parameters of the face features typically are between -1 and 1 or at least within lower digits, our landmark matching cost only produced extremely small cost values in the $10^{-}7$ range.

The reason for this was that we used meters as the scale of our world and thus the distances to the landmarks would only be a few mm. This also made us notice that our matching cost was dependent on the scale of the scene, eg. if we have a scene where the face is 20 units wide compared to a scene where the face is 0.2 units wide. This would lead to drastically different focus on the parameter optimization versus the regularization leading to the face staying extremely close to the mean shape on one case while completely overfitting in the other. To solve this, we decided to divide the cost by the current scale of the face model so it should be scale invariant.

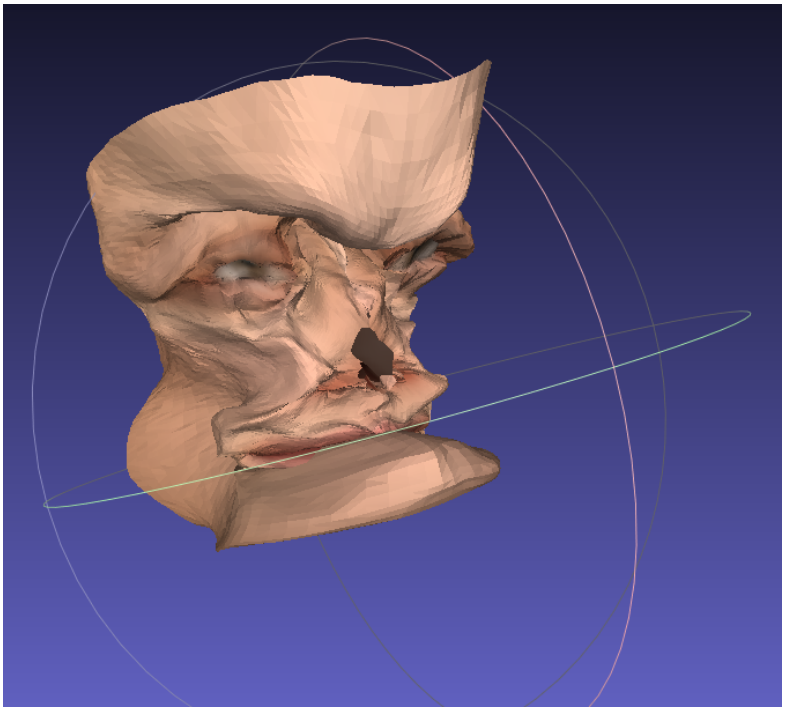

Figure 7: Fitting BFM models with too little regularization turns out to be a valid way to generate shitty digital art. Definitely the next thing for NFTs

Group member. Apart from that, Dave decided to leave our group since he didn't have enough time to properly contribute.

3 Plan

We already have a rough setup for the parameter estimation of the Basel Face Model that we want to improve on until the project deadline. In particular, we want to add some kind of dense term to try to match a subset of the vertices of the BFM to the

surface of the depth image in an ICP-like fashion. We hope that this will enable the BFM to more closely match the actual face shape while the sparse landmark-term keeps the face aligned and improves the speed. We are currently considering to apply this only to a subsection of the vertices of the face mesh, since the mesh is rather dense and we hope to be able to speed up the matching this way.

1. Detect face in image (done)
2. Detect landmarks in face image (done)
3. Project the detected landmarks into world space (similar to below)
4. Reconstruct point cloud of the relevant parts of the image (done)
5. Extract the corresponding 68 landmarks from Basel Face Model (done)
6. Use the 3D positions of the landmarks for Procrustes alignment to estimate the pose of our point cloud (done)
7. Use landmark 3D positions in a sparse term for optimizing the parameter of face shape, and expression, etc. (done)
8. Use the BFM face vertices in a dense term for optimizing the parameter of face shape, and expression, etc (In progress)

References

[1] Christos Sagonas. ibug - resources - Facial point annotations, 2016.